CLAIMS

1-18 (cancelled)

19. (New) A transaction based constraint enforcer for a database system,
for enforcing a set of constraints that governs the integrity of information stored in the database system, said enforcer being arranged to delay constraint checks until the end of a transaction by creating a check stack during the course of the transaction and executing entries on the check stack at the end of the transaction,
said constraint enforcer comprising
- a stack maker module, arranged for creating and updating said check stack, said stack maker module being operatively connected to a runtime module in the database system and arranged to receive data from said runtime module,
said stack maker module being further arranged
- to perform a delete process on said check stack when said stack maker module is called upon as a result of a Delete or Update Data Manipulation Language operation,
- to perform an insert process on said check stack when said stack maker module is called upon as a result of an Insert or Update Data Manipulation Language operation, and
- to perform said delete process followed by said insert process when said stack maker module is called upon as a result of an Update Data Manipulation Language operation.

20. (New) Constraint enforcer according to claim 19, further comprising
an enforcer module, arranged to receive check data from the check stack, to process the check data received from the check stack, and to provide resulting data to the runtime module.

21. (New) Constraint enforcer according to claim 20, wherein said constraints are stored in a conceptual rules module, comprising rules for prescribing permitted states and transitions that the database can undertake, said conceptual rules module being operatively connected to said stack maker module, wherein said stack maker module is arranged to retrieve constraints from said conceptual rules module.

22. (New) Constraint enforcer according to claim 19,
wherein said check stack is stored on persistent or volatile memory.

23. (New) Method for enforcing a set of constraints that governs the integrity of information stored in a database system, comprising the step of
delaying constraint checks until the end of a transaction by creating a check stack during the course of the transaction and executing entries on the check stack at the end of the transaction, the method further comprising the following steps, performed by a stack maker module operatively connected to a runtime module in said database system:

- receiving data from said runtime module,
- creating and updating said check stack,
- performing a delete process on said check stack when said stack maker module is called upon as a result of a Delete or Update Data Manipulation Language operation,
- performing an insert process on said check stack when said stack maker module is called upon as a result of an Insert or Update Data Manipulation Language operation, and
- performing said delete process followed by said insert process when said stack maker module is called upon as a result of an Update Data Manipulation Language operation.

24. (New) Method according to claim 23, further comprising the following steps, performed by an enforcer module:
receiving check data from the check stack,
processing the check data received from the check stack, and
providing resulting data to the runtime module.

25. (New) Method according to claim 24, wherein said constraints are stored in a conceptual rules module, comprising rules for prescribing permitted states and transitions that the database can undertake, further comprising the step of retrieving by said stack maker module constraints from said conceptual rules module.

26. (New) Method according to claim 23,
wherein said check stack is stored on persistent or volatile memory.

27. (New) A database system, comprising
an application program interface, providing a two-way message interface to a user application program,
a runtime module, operatively connected to the application program interface,
a storage engine module, operatively connected to the runtime module,
a data storage, operatively connected to the storage engine module, and
a transaction based constraint enforcer, for enforcing a set of constraints that governs the integrity of information stored in the database system, said enforcer being arranged to delay constraint checks until the end of a transaction by creating a check stack during the course of the transaction and executing entries on the check stack at the end of the transaction,
the constraint enforcer further comprising a stack maker module, arranged for creating and updating said check stack, said stack maker module being operatively connected to said runtime module in the database system and arranged to receive data from said runtime module, said stack maker module being further arranged for

- performing a delete process on said check stack when said stack maker module is called upon as a result of a Delete or Update Data Manipulation Language operation,
- performing an insert process on said check stack when said stack maker module is called upon as a result of an Insert or Update Data Manipulation Language operation, and
- performing said delete process followed by said insert process when said stack maker module is called upon as a result of an Update Data Manipulation Language operation.

28. (New) System according to claim 27, wherein said constraint enforcer further comprises an enforcer module, arranged to receive check data from the check stack, to process the check data received from the check stack, and to provide resulting data to the runtime module.

29. (New) System according to claim 28,
wherein said constraints are stored in a conceptual rules module, comprising rules for prescribing permitted states and transitions that the database can undertake, said conceptual rules module being operatively connected to said stack maker module, said stack maker module being arranged to retrieve constraints from said conceptual rules module.

30. (New) System according to claim 27,
wherein said check stack is stored on persistent or volatile memory.